

Egon Börger (Pisa)

Reachability by Shortest Paths in a Graph

Illustrating Stepwise ASM Refinements

Dipartimento di Informatica, Università di Pisa
boerger@di.unipi.it

See Modeling Companion Ch. 4.3 (refinement variations of the PROXY programming pattern) and AsmBook Ch.3.2¹

¹ Figures © 2003 Springer Berlin-Heidelberg, reused with permission.

The problem

Given a directed graph:

- a (finite) set $Node$
- a set $Edge$
- a distinguished node $source$

Design algorithms to compute:

- the set of nodes n which are reachable from $source$
- for each node n the 'shortest' path from $source$ to n (wrt a path measure given by the weight of edges)

Verify the algorithms proving their correctness and other properties of interest.

A sequence of refinement steps

- Computing Reachability Set: SHORTESTPATH_0 (ground model)
- Wave Frontier Propagation: SHORTESTPATH_1
- Nodewise Frontier Propagation to Neighborhood: SHORTESTPATH_2
- Nodewise and Edgewise Frontier Propagation to Neighbors:
 SHORTESTPATH_3
- Queue and Stack Implementation of Frontier and Neighborhoods:
 SHORTESTPATH_4
- Introducing abstract weights for measuring paths and computing shortest paths: SHORTESTPATH_5 (Moore's algorithm)
- Instantiating data structures for measures and weights: a C^{++} program

Computing Graph Reachability Set

The problem:

- visit once every node which is reachable from *source*
- do not revisit nodes that have already been *visited*, so that the procedure terminates for finite graphs

Solution idea:

- starting at *source*, move along edges to reach neighbor nodes and label every reached node as *visited*
- proceed in waves, pushing in each step the visited nodes one edge further *without revisiting nodes* which have already been labeled as *visited*

From English to a mathematical model: SHORTESTPATH₀

Initial state: only *source* is labeled as *visited*

SHORTESTPATH₀ =

forall $u \in \textit{visited}$ **forall** $v \in \textit{neighb}(u)$ -- wave propagation

if $v \notin \textit{visited}$ **then** $\textit{visited}(v) := \textit{true}$

where

$\textit{neighb}(u) = \{v \mid (u, v) \in E\}$

- **Correctness Property:** Each node which is reachable from *source* is exactly once labeled as visited.
- Proof of existence claim: induction on the length of the paths starting at source.
 - induction basis: by initialization assumption
 - induction step: by applying SHORTESTPATH₀ rule once more
- Proof of uniqueness property: rule guard ensures that no node which has already been *visited* is relabeled.

Proving termination for SHORTESTPATH_0

Termination Property: SHORTESTPATH_0 terminates for finite graphs: started in the initial state, it reaches a state in which there is no longer any edge $(u, v) \in E$ where u is labeled as *visited* but v is not.

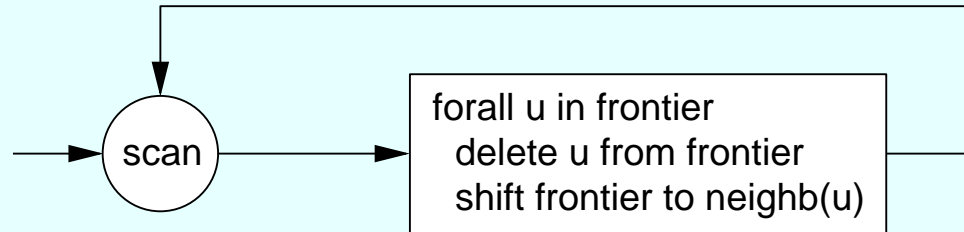
- Proof: by each step of SHORTESTPATH_0 the (assumed to be finite) set of reachable nodes which have not yet been *visited* decreases.

NB. The refinement steps should preserve Correctness and Termination.

First refinement: identify *frontier* of wave propagation

frontier = set of nodes 'last labeled' as *visited* (here: in the last step)

Initially *frontier* = {*source*}



SHORTESTPATH₁ = **forall** $u \in \textit{frontier}$ -- restrict to 'last labeled'

SHIFTFRONTIERTONEIGHB(u)

DELETE($u, \textit{frontier}$) -- not any more 'last labeled'

where

SHIFTFRONTIERTONEIGHB(u) =

forall $v \in \textit{neighb}(u)$ **do** SHIFTFRONTIERTO(v)

SHIFTFRONTIERTO(v) = **if** $v \notin \textit{visited}$ **then**

$\textit{visited}(v) := \textit{true}$ INSERT($v, \textit{frontier}$) -- v gets 'last labeled'

Proving refinement correctness

Claim: SHORTESTPATH_1 is a correct refinement of SHORTESTPATH_0 .

Proof. Show by induction on runs that the **labeling steps** of SHORTESTPATH_0 and of SHORTESTPATH_1

- which label some neighbor of some u as *visited*

are in 1-1 correspondence and perform the same labelings.

Induction basis $t = 1$: both machines perform one labeling step (if any) and label exactly the nodes in $\text{neighb}(\text{source})$ as *visited*

- since by initialization $\text{frontier} = \{\text{source}\}$

Refinement correctness proof (Cont'd)

Induction step $t \Rightarrow t + 1$:

- Consider any $u \in \textit{frontier}$.
 - **if** SHORTESTPATH_1 can make a labeling step with $\text{SHIFTFRONTIERTONEIGHB}(u)$
 - **then** SHORTESTPATH_0 can make a labeling step for $\textit{neighb}(u)$ so that the same nodes are labeled as newly *visited*.
- In step $t + 1$, SHORTESTPATH_0 applies labeling only to $u \in \textit{frontier}$.
 - Proof. For every $u \notin \textit{frontier}$: if u has been *visited* by SHORTESTPATH_0 in a step before step t , then all its *neighbors* have been *visited* in the next step of SHORTESTPATH_0 . Therefore SHORTESTPATH_0 does not revisit them in step $t + 1$.

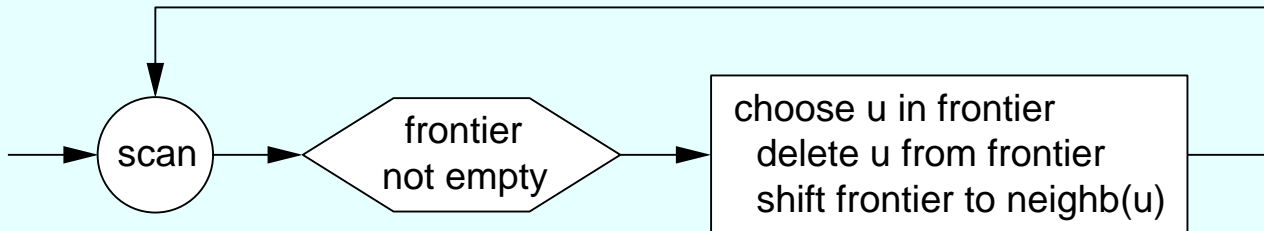
A non-empty step of SHORTESTPATH_1 which is not a labeling step may be (only) the last one: it empties *frontier*.

Therefore *Correctness and Termination* properties are *preserved*.

Second refinement: implementing 'forall'

Idea: nodewise frontier propagation, implementing **forall** by **choose**

- Non-deterministic scheduling (to keep design space open)
- Later refinements specify constraints on *selection* fct to guarantee properties of interest (e.g. fairness to yield completeness of node visits)



SHORTESTPATH₂ =

if $frontier \neq \emptyset$ **then** **choose** $u \in frontier$ -- replacing **forall**

SHIFTFRONTIERTONEIGHB(u) DELETE($u, frontier$)

last labeled in *frontier* is refined to mean any *visited* node u to which SHIFTFRONTIERTONEIGHB(u) has not yet been applied

Refinement correctness for nodewise frontier propagation

Simulation Lemma. SHORTESTPATH_2 runs with *breadth-first nodewise frontier propagation* simulate SHORTESTPATH_1 runs.

- In other words, SHORTESTPATH_2 with breadth-first nodewise frontier propagation is a correct refinement of SHORTESTPATH_1 .

Proof: One SHORTESTPATH_1 step, applied to a *frontier*, corresponds to the segment of SHORTESTPATH_2 steps which choose successively all and only the nodes from this *frontier*

- applying the same $\text{SHIFTFRONTIERTONEIGHB}(u)$.

This is called a $(1, m)$ -refinement with various m , depending on the size m of the neighborhoods which dynamically determine the frontier.

Relating frontier propagation in $\text{SHORTESTPATH}_{1/2}$ runs

Slow Down Lemma. For maximal SHORTESTPATH_i runs ($i = 1, 2$), i.e. where each applicable rule is eventually applied, the following holds:

1. Claim 1. For each step t and each $u \in \text{frontier}_t(\text{SHORTESTPATH}_2)$ there exists a $t' \leq t$ such that $u \in \text{frontier}_{t'}(\text{SHORTESTPATH}_1)$.
2. Claim 2. For each step t and each $u \in \text{frontier}_t(\text{SHORTESTPATH}_1)$ there exists a $t' \geq t$ such that $u \in \text{frontier}_{t'}(\text{SHORTESTPATH}_2)$.

Here we denote by exp_t the value of exp in the state reached by t steps. An index $i \in \{1, 2\}$ in $\text{exp}_t(i)$ refers to the value in a state of SHORTESTPATH_i .

Corollary. SHORTESTPATH_i for $i = 1, 2$ label the same nodes as *visited*, once. Thus, the refinement preserves *Correctness and Termination*.

Slow Down Lemma: Proof by induction on runs

- $t = 0$: SHORTESTPATH_i both have $\text{frontier}_0(i) = \{\text{source}\}$
- $t \Rightarrow t + 1$:
 - ad claim 1:
 - Case 1. Let $v \in \text{frontier}_t(2)$. Then by induction hypothesis $v \in \text{frontier}_{t'}(1)$ for some $t' \leq t$.
 - Case 2. Let $v \in \text{frontier}_{t+1}(2) \setminus \text{frontier}_t(2)$.
Let $u \in \text{frontier}_t(2)$ be the element chosen by step $t + 1$ of SHORTESTPATH_2 . By case 1, $u \in \text{frontier}_{t'}(1)$ for some $t' \leq t$.
Then after the next step of SHORTESTPATH_1 , namely $t' + 1 \leq t + 1$, each element of $\text{neighb}(u)$ is *visited*, including $v \in \text{neighb}(u)$.
Hence, either v has been labeled as *visited*(1) already before step $t' + 1$, so that $v \in \text{frontier}_{t''}(1)$ for some $t'' \leq t'$, or v is ‘last labeled’ as *visited* by step $t' + 1$ of SHORTESTPATH_1 , which implies $v \in \text{frontier}_{t'+1}(\text{SHORTESTPATH}_1)$.

Slow Down Lemma: Proof by induction on runs (Cont'd)

- ad claim 2: By definition of SHORTESTPATH_1 , the following equation holds:

$$\text{frontier}_{t+1}(1) = \bigcup_{u \in \text{frontier}_t(1)} \text{neighb}(u) \setminus \text{Visited}_t(1).$$

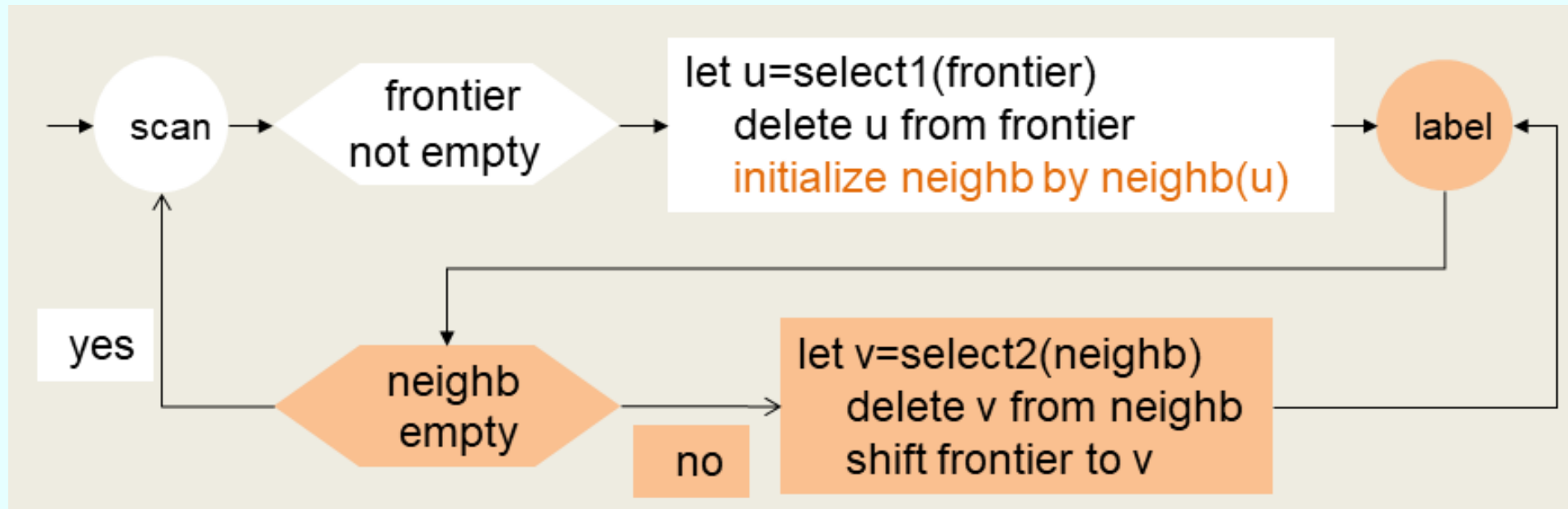
- For every $u \in \text{frontier}_t(1)$ the induction hypothesis implies $u \in \text{frontier}_{t'}(2)$ for some $t' \geq t$.
- Let $u \in \text{frontier}_{t'}(2)$ be chosen by SHORTESTPATH_2 in step $t' + 1$. For each $v \in \text{neighb}(u) \setminus \text{Visited}_t(1)$ this step yields $v \in \text{frontier}_{t'+1}(2)$.

Third refinement: iterative implementation of 'forall'

Idea: edgewise frontier propagation, implementing **forall** in $\text{SHIFTFRONTIERTONEIGHB}(u)$ by an iterating submachine

- initialize $\text{neighb} = \text{neighb}(u)$ and then select one by one nodes v of neighb to edgewise $\text{SHIFTFRONTIERTO}(v)$ until $\text{neighb} = \emptyset$

$\text{SHORTESTPATH}_3 =$



Refinement correctness for edgewise frontier propagation

- each iteration segment of successive SHORTESTPATH_3 steps which
 - first choose an $u \in \text{frontier}$
 - then choose successively all and only the neighbors v of u
 - to apply $\text{SHIFTFRONTIERTO}(v)$
- corresponds and is equivalent to one SHORTESTPATH_2 step
 - which applies to the same $u \in \text{frontier}$ in one $\text{SHIFTFRONTIERTONEIGHB}(u)$ step simultaneously $\text{SHIFTFRONTIERTO}(v)$ for all $v \in \text{neighb}(u)$

This is an example of a $(1, \text{many})$ refinement, with u -dependent values of many , defined by the graph structure:

$$\text{many} = 1 + \text{outFan}(u)$$

Corollary. The refinement of SHORTESTPATH_2 to SHORTESTPATH_3 preserves *Correctness and Termination*.

Fourth refinement: implementing data structures

SHORTESTPATH₄ refines two data structures:

- *frontier* is refined by a *queue*: $select1 = fst$, DELETE at one end and APPEND at the other end
- *neighb* is refined to a *stack*: $select2 = top$, DELETE = *pop*

This is an example of a **pure data refinement**:

- the steps in the runs are in 1-1-correspondence

Refinement Correctness of this 1-1 refinement boils down to the correctness of the well-known set implementations by queues resp. stacks.

Corollary. The data refinement of SHORTESTPATH₃ to SHORTESTPATH₄ preserves *Correctness and Termination*.

Introducing weights to measure paths from *source* to *u*

Extend a given edge $weight: E \rightarrow \mathbb{R}^+$ to a **path weight** as follows:

$$weight(\epsilon) = 0, weight(pe) = weight(p) + weight(e)$$

$$minWeight(u) = inf\{weight(p) \mid p \text{ is a path from } source \text{ to } u\}$$

NB. Instead of \mathbb{R}^+ , any well-founded partial order $(M, <)$ of path measures works which has the following properties:

- there are a smallest and a largest element 0 resp. ∞
- any $m, m' \in M$ have an *infimum* (greatest lower bound)
- adding edge weight to path measures is monotonic wrt path measures and distributive wrt *inf*, i.e. for each $m, m' \in M$ and edge weight w :
 - $m < m'$ implies $m + w < m' + w$
 - $inf(X) + w = inf\{x + w \mid x \in X\}$

Fifth refinement: compute minimal path from *source* to *u*

Goal: when visiting *u* from *source*, compute also a minimal path, i.e. of minimal $\text{minWeight}(u)$, along the (possibly multiple) paths

- by *successive approximations of an upper bound* $\text{upbd}: \text{NODE} \rightarrow \mathbb{R}$
- starting with $\text{upbd}(u) = \infty$ for every node *u*, except $\text{upbd}(\text{source}) = 0$

Idea: Refine `SHIFTFRONTIERTO`, along an edge *e* from *u* to *v*

- trying to lower $\text{upbd}(v)$ to $\text{upbd}(u) + \text{weight}(e)$

Problem: *feature interaction* of conflicting (not purely incremental) requirements, namely:

- each node is visited only once
- compute a minimal path from source to each reachable node by stepwise improving approximations, possibly discovering a shorter path upon revisiting the node

Conflict resolution: revisit nodes if it improves their $upbd$

Conflict resolution: Each time $upbd(v)$, for a path from *source* to v , *CanBeLoweredBy* a path going through an edge (u, v) from an already *visited* neighbor $u \in frontier$, v is INSERTed into *frontier*:

- When $v \neq source$ is *visited* for the first time, say via an already *visited* neighbor node $u \in frontier$ (so that $upbd(u) < \infty$), its $upbd(v) = \infty$ *CanBeLoweredBy* updating it using $upbd(u) + weight(u, v)$.
- When $upbd(v) < \infty$ (so that v has already been *visited*) but $upbd(v)$ *CanBeLoweredBy* a path going through a neighbor node $u \in frontier$, v is ‘revisited’
 - meaning that it is INSERTed once more into *frontier*.

Refinement implementing conflict resolution

SHORTESTPATH₅ is SHORTESTPATH₄ refined as follows:

- Add $currSource := u$ to $neighb := neighb(u)$ initialization

- SHIFTFRONTIERTO(v) =

- if** $v \notin visited$ **then** -- upon first visit $upbd(v) = \infty$ holds

- $visited(v) := true$

- INSERT($v, frontier$)

- LOWERUPBD($upbd(v), (currSource, v)$) -- yields $upbd(v) < \infty$

- because $u \in frontier$ implies $upbd(u) < \infty$

- else**

- if** $CanBeLoweredBy(upbd(v), (currSource, v))$ **then**

- LOWERUPBD($upbd(v), (currSource, v)$)

- INSERT($v, frontier$) -- neighbors may have to LOWERUPBD

Meaning of LOWERUPBD

CanBeLoweredBy($bd, (u, v)$) **iff**

$upbd(u) + weight(u, v) < bd$ -- Dijkstra's algorithm where $M = \mathbb{R}$

$bd \not\leq upbd(u) + weight(u, v)$ -- Moore's algorithm

$LOWERUPBD(bd, (u, v)) =$

$bd := upbd(u) + weight(u, v)$ -- Dijkstra

$bd := \inf\{bd, upbd(u) + weight(u, v)\}$ -- Moore

Remark. A further refinement step could restrict *frontier* to a *priority queue*, selecting nodes with least *upbd*:

$u = select1(frontier)$ **iff forall** $v \in frontier$ $upbd(u) \leq upbd(v)$

Refinement is labeling correct and complete

- **Completeness:** by definition, SHORTESTPATH_5 is a purely incremental extension, also called *conservative refinement*, of SHORTESTPATH_4 .
 - In fact, each SHORTESTPATH_4 step corresponds to a step of SHORTESTPATH_5 with equivalent labeling:
 - *select1*-steps choosing an u for the first time in *frontier*
 - *select2*-steps with a first-visit application of SHIFTFRONTIERTO
- **Correctness:** for every pair (r_4, r_5) of corresponding SHORTESTPATH_i runs ($i = 4, 5$),
 - i.e. runs started in the same initial state and with (where corresponding) same *selections* projecting from r_5 a) *select1*-steps which choose an u that is for the first time in *frontier*, and b) *select2*-steps with a first-visit application of SHIFTFRONTIERTO (not considering the LOWERUPDB submachine) yields r_4 .

Termination of SHORTESTPATH₅

Since elements may be reinserted into *frontier*, to prove the termination property it has to be shown that eventually *frontier* becomes empty.

Each time in mode *scan* an element $u \in \textit{frontier}$ is selected, at the end of the iteration, when *mode* becomes again *scan*, the following holds:

- either *frontier* is decreased by 1
 - namely if for none of u 's neighbors v
 $\textit{CanBeLoweredBy}(\textit{upbd}(v), (u, v))$ so that the sum of $\textit{upbd}(v)$
of u 's neighbors v remains unchanged
- or the number of u 's neighbors v with $\textit{upbd}(v) = \infty$ or the sum of $\textit{upbd}(v) < \infty$ of u 's neighbors v is decreased
 - which can happen only finitely often

Correctness of Shortest-Path-Property for Moore's algorithm

Theorem. When SHORTESTPATH_5 terminates, for every u holds:

$$\text{minWeight}(u) = \text{upbd}(u)$$

The proof follows from two lemmata:

- **Lemma 1.** $\text{minWeight}(u) \leq \text{upbd}(u)_t$ holds for each u after each step t .
- **Lemma 2.** When SHORTESTPATH_5 terminates, $\text{upbd}(u) \leq \text{weight}(p)$ holds for every path p from *source* to u .

In fact, let t be the last step of SHORTESTPATH_5 . Then

$$\begin{aligned} \text{minWeight}(u) &\leq \text{upbd}(u)_t && \text{-- by Lemma 1} \\ &\leq \text{weight}(p) && \text{-- by Lemma 2 for each source-to-}u \text{ path } p \end{aligned}$$

Thus, $\text{upbd}(u)_t$ is a lower bound of $\text{weight}(p)$ for every *source-to- u* path p , therefore $\text{upbd}(u)_t \leq \text{minWeight}(u)$, the greatest such bound. Hence $\text{minWeight}(u) = \text{upbd}(u)_t$.

Proof of Lemma 1 ($\min Weight(u) \leq upbd(u)_t$) by induction

- $t = 0$: claim holds by definition of $upbd(source) = 0$ and $upbd(u) = \infty$ for each $u \neq source$
- if in step $t + 1$ $upbd(v)$ is updated, then to $\inf\{upbd(v)_t, upbd(u)_t + weight(u, v)\}$. Since $\min Weight(v)$ is a lower bound for both values in the set (see below), it is \leq their greatest lower bound.
 - $\min Weight(v) \leq upbd(v)_t$ holds by ind.hyp.
 - $\min Weight(v) \leq \min Weight(u) + weight(u, v)$ (see below) and ind.hyp. $\min Weight(u) \leq upbd(u)_t$ imply the claim.
 - $\min Weight(v)$
 - $=_{Def} \inf\{weight(p) \mid p \text{ path from } source \text{ to } v\}$
 - $\leq \inf\{weight(p.(u, v)) \mid p \text{ path from } source \text{ to } u\}$
 - $=_{Def} \inf\{weight(p) + weight(u, v) \mid p \text{ path from } source \text{ to } u\}$
 - $=_{Distr} \inf\{weight(p) \mid p \text{ path from } source \text{ to } u\} + weight(u, v)$
 - $=_{Def} \min Weight(u) + weight(u, v)$

Proof of Lemma 2 ($upbd(u) \leq weight(p)$ upon termination)

Induction on path length t :

- $t = 0$: claim follows from $upbd(source) = 0 = weight(\epsilon)$
- Let $p.(u, v)$ be any path of length $t + 1$.
 - $upbd(v) \leq upbd(u) + weight(u, v)$
 - otherwise LOWERUPBD($upbd(v), u$) could fire
 - $upbd(u) \leq weight(p)$ by ind.hyp.

Therefore

$$\begin{aligned} upbd(v) &\leq upbd(u) + weight(u, v) \\ &\leq weight(p) + weight(u, v) // \text{ by monotonicity} \\ &= weight(p.(u, v)) // \text{ by definition of path } weight \end{aligned}$$

Refinement to C^{++} code

- It remains to instantiate data structures for measures and weights
- See: K. Stroetmann: The Constrained Shortest Path Problem: A Case Study in Using ASMs.
 - J. of Universal Computer Science 3 (4), 1997.

References

- *E. F. Moore: The Shortest Path Through a Maze.* Proc. Intern. Symp. on the Theory of Switching, Part II, Vol. 30 of The Annals of the Computation Laboratory of Harvard University, Cambridge, MA, 1959, Harvard University Press.
- *K. Stroetmann: The Constrained Shortest Path Problem: A Case Study in Using ASMs.* In: J. of Universal Computer Science 3 (4), 1997.
- E. Börger and R. Stärk: Abstract State Machines. Springer 2003. See Ch.3.2.1 for a correctness proof for Dijkstra's algorithm `SHORTESTPATH5`.
- E. Börger and A. Raschke: Modeling Companion for Software Practitioners. Springer 2018
<http://modelingbook.informatik.uni-ulm.de>

Copyright Notice

It is permitted to (re-) use these slides under the CC-BY-NC-SA licence

<https://creativecommons.org/licenses/by-nc-sa/4.0/>

i.e. in particular under the condition that

- the original authors are mentioned
- modified slides are made available under the same licence
- the (re-) use is not commercial